

Open in app ↗



Search



A step by step guide: System design Interviews



Suresh Podeti

3 min read · Sep 29, 2023



Listen



Share

... More

Photo by [Am lie Mourichon](#) on [Unsplash](#)

Introduction

System design problems are not straightforward. We don't have a universal formula that we can use for all design problems. However, we can use a high-level common strategy to set the tone for a good solution to any design problem.

We call this the **RESHADED** approach. Generally, all our design problems are solved by keeping this strategy in mind. RESHADED is a guideline that we'll use to resolve different design problems. Although there's no such thing as a one-size-fits-all solution.

Solution offered through the RESHADED approach will be systematic and thoughtful.

RESHADED

R(equirements)

During this step, we gather all the requirements of the design problem and define its scope. Requirements include understanding what the service is, how it works, and what its main features are. Our goal in this step is to gather the functional and non-functional requirements of a service we are about to design.

E(stimation) or back-of-the-envelope estimation

This step estimates the **resources** required to provide the service to a defined number of users. By resources, we mean the hardware or infrastructural resources.

Estimations are important because they help us understand the scale of the system we'll design. We'll make key decisions based on the estimate. For example, we'll decide what type of database to use for storing our data, which data structure will give optimal performance, and so on.

S(torage schema)

This step involves articulating our data model — that is, we define which tables we need and what type of fields are part of each table.

H(igh-level design)

This step involves identifying the main components and building blocks we'll use to design our desired system. We do this by getting inspiration from our functional and non-functional requirements.

This is considered the first step toward the complete design of our system and therefore requires further iteration and improvement. Primarily, this section will focus on fulfilling the functional requirements.

A(PI design)

The goal in this phase is to build interfaces for our service. Using these interfaces, users can call various services within our system. These interfaces are in the form of

API calls and are generally a translation of our functional requirements.

Detailed design)

The detailed design starts by recognizing the limitations of the high-level design. We'll capitalize on these limitations to evolve our design. During this step, we'll finalize our design by mentioning all the components and building blocks that we'll use.

We also define the workflow of our design and its usage of different technologies. The detailed design aims to fulfill the functional as well as non-functional requirements of the problem.

Evaluation)

This step will measure the effectiveness of our solution. In other words, we justify how our design fulfills the functional and non-functional requirements.

Distinctive component/feature)

At the start of this lesson, we discussed how no one-size-fits-all solution exists. This step is to identify a unique aspect for each design problem and discuss it. For example, the Uber design problem has payment service and fraud detection as its unique feature. In contrast, Google Docs has concurrency control, which is required when different users want to edit the same section of a document simultaneously.

[System Design Interview](#)[Software Architect](#)[Software Architecture](#)[Distributed Systems](#)[System Design Concepts](#)[Edit profile](#)

Written by Suresh Podeti

1.2K Followers